Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Static Hand Pose Recognition on Depth Images

(Reconocimiento de pose estática de manos en imágenes de profundidad)

Documento de tesis sometido a consideración para optar por el grado académico de Maestría en Electrónica con Énfasis en Procesamiento Digital de Señales

Marco Madrigal Solano

Cartago, 15 de mayo, 2015

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Marco Madrigal Solano

Cartago, 15 de mayo, 2015

Céd: 1-1315-0600

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Tesis de Maestría
Tribunal Evaluador

Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Alex Singh

Dr. Alexander Singh Alvarado
Profesor Lector

Dr. Juan Luis Crespo Mariño
Profesor Lector

Dr. Pablo Alvarado Moya
Profesor Asesor

Los miembros de este Tribunal dan fe de que la presente tesis de maestría ha sido aprobada y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 15 de mayo de 2015

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Tesis de Maestría
Tribunal Evaluador
Acta de Evaluación

Tesis de maestría defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de maestría, del Instituto Tecnológico de Costa Rica.

Estudiante: Marco Emilio Madrigal Solano

Nombre del Proyecto: *Static Hand Pose Recognition on Depth Images*

Miembros del Tribunal

Alex Singh

Dr. Alexander Singh Alvarado
Profesor Lector

Dr. Juan Luis Crespo Mariño
Profesor Lector

Dr. Pablo Alvarado Moya
Profesor Asesor

Los miembros de este Tribunal dan fe de que la presente tesis de maestría ha sido aprobada y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Nota final de la Tesis de Maestría: 100

Cartago, 15 de mayo de 2015

Resumen

Con el aumento del uso de nuevas tecnologías en actividades diarias, la demanda de sistemas de interacción humano-máquina (HCI por sus siglas en Inglés) ha incrementado. Sistemas de reconocimiento de pose de manos han sido ampliamente explorados para dicha tarea debido a su operación intuitiva para usuarios inexpertos. Sin embargo, el reconocimiento de pose de manos basados en visión es un problema extremadamente desafiante debido a la dinámica de la mano, la cual posee un gran número de grados de libertad que la hacen difícil de estimar y conlleva a problemas adicionales como la auto oclusión. Con el desarrollo se sistemas de visión confiables y asequibles para el usuario común como el Microsoft Kinect©, las imágenes de profundidad se han convertido en una herramienta útil para el reconocimiento de partes del cuerpo y por tanto, para el reconocimiento de manos.

En esta tésis se propone un sistema de clasificación de poses de mano estáticas basado solamente en imágenes de profundidad y considerando una perspectiva de vista superior. No se establecen restricciones adicionales a la posición de la mano en la escena, lo cual permite a otros objetos estar más cerca de la cámara que la mano. Un conjunto de datos sintéticos es generado para cuatro posturas de la mano (*abierta*, *apuntando*, *puño* y *pinza*). El diseño propuesto es dividido en dos etapas de procesamiento: segmentación de la mano y clasificación de la pose de la mano. La etapa de segmentación de la mano utiliza un bosque de decisión aleatorio (RDF por sus siglas en Inglés) para una clasificación a nivel de pixel de las imágenes de profundidad, segmentando la mano en cuatro regiones: *brazo*, *palma* y *dedos*. La clasificación de la pose de la mano se lleva a cabo utilizando un conjunto definido de características visuales de las regiones segmentadas. Siete características visuales son evaluadas en términos de su precisión de clasificación. Dos tipos de clasificadores son entrenados para la estimación de la pose: bosques de decisión aleatorios y máquinas de soporte vectorial (SVM por sus siglas en Inglés) para propósitos de evaluación. La implementación propuesta provee un 91% de precisión en la casificación de las poses de mano utilizadas con el conjunto de datos generado.

Palabras clave: RDF, SVM, imágenes de profundidad, pose de mano, momentos de Hu

Abstract

With the increasing usage of new technologies in common daily activities, the demand of efficient human-computer interaction (HCI) systems increases. Hand pose recognition systems have been widely explored for such task due to its intuitive operation for non experienced users. However, vision-based hand pose recognition is a extremely challenging problem due to the dynamics of the hand, which poses a large amount of degrees of freedom that makes it difficult to estimate and carries out additional problems such as self occlusion. With the development of reliable and consumer affordable vision systems such as the Microsoft Kinect©, depth imaging has become a useful tool on body parts recognition and thus, for hand recognition.

This thesis proposes a static hand pose classification system based on depth images only and considering a top view perspective. No additional constrains to the hand position on the scene are imposed, which allows background objects to be closer to the camera than the hand itself. A synthetically generated data set of four hand postures (*open*, *pointing*, *fist* and *pinch*) is used. The proposed design is divided in two processing stages: hand segmentation and hand pose classification. The hand segmentation stage uses a random decision forest (RDF) for per-pixel classification of the depth images, segmenting the hand in *arm*, *palm* and *fingers* regions. Hand pose classification is then performed using a defined set of visual features from the labeled blobs. Seven visual features are evaluated in terms of classification accuracy. Two types of classifiers are trained for the pose estimation: random decision forests and support vector machine (SVM) for evaluation purposes. The system proposed provides a 91% of classification accuracy for the defined hand poses on the generated data.

Keywords: RDF, SVM, depth images, hand pose, Hu moments

to my family

Acknowledgements

First of all, I thank my family and friends for all the support given in those years, I would never have made it without them.

Additionally, I would like to thank my supervisor Dr. Pablo Alvarado Moya and the Dr. Alexander Singh Alvarado for their guidance and advice during the development of my thesis, providing me a valuable experience and knowledge.

Finally, I would like to thank Ridgerun for all its support during the process and for allowing me the space necessary to finalize the thesis presented in this document. Its understanding and support were a vital component that enable me to hold a master's degree today.

Marco Madrigal Solano

Cartago, 15 de mayo, 2015

Contents

List of Figures

List of Tables

Glossary

Abreviaciones

IR	Infrared
RDF	Random Decision Forest
RNO	Randomized Node Optimization
ROI	Region Of Interest
SVM	Support Vector Machine

Chapter 1

Introduction

Human-computer interaction (HCI) has taken a key role in the modern society due to the latest technology advances that have increased the usage of computer based systems in common daily activities. Within the whole range of applications using vision based systems, hand controlled systems are widely extended due to their easy use and intuitive nature. Hands provide a dexterous functionality in communication and manipulation that makes them a perfect tool for any interactive application[11] compared with other approaches such as face or voice control. The usage of hand recognition-based systems covers a wide range of applications such as virtual reality and simulation [3, 29], mobile devices [30], sign language recognition [29], musical gestures recognition [7], among others.

Despite hands are acknowledged as an effective interaction tool, they carry out a set of challenges in regards with their detection using computer vision systems[11]:

- **Dimensionality**: With approximately 27 degrees of freedom, the hand pose is highly difficult to estimate compared with simpler recognition systems such as body pose recognition [8, 21, 11].
- **Self-occlusion**: Due to the amount of degrees of freedom, the hand suffers of self-occlusion, i.e., on a variety of positions and points of view from the camera, some sections of the hand are occluded by others avoiding its direct estimation with vision computer systems[26, 17, 8].
- **Noise**: The range sensors available nowadays present a relatively high level of noise in the depth information (e.g. Gaussian noise with standard deviation of $1cm$ at $1m$ distance for PrimeSense sensors)[26, 11, 8].
- **Scene variability**: Wide range of applications implies that the hand will move among arbitrary scenes with background objects, illumination camera positions and orientations.
- **Processing speed**: In order to provide a hand recognition system that meets the natural interaction requirements, it is necessary to keep latency times low while

processing frame rates over 20fps. However most of the available proposals for hand recognition require a powerful system to run on such timing conditions, making them infeasible for the market.

With the introduction of the depth sensor at an accessible cost for the common user, most of those challenges discussed above have been resolved or treated on a simpler way providing a new field of research for hand recognition. Despite all the improvements provided by depth sensors for image processing application and more specific for hand recognition, usually they are used together with color sensors for complementary image processing[23, 10, 22]. Additionally, some common assumptions such as distance to the camera and location constrains are usually taken when dealing with depth images, to simplify the whole processing pipeline.

This thesis is part of a global concept to approach the estimation of hand joints of multiple hands in a single scene from a top view, using depth images only. The general proposal is divided in three major processing stages as depicted in figure 1.1: the first stage aims to the detection of one or more hands on a depth image scene and provide the approximated location. The second stage uses the input information to segment the hand region and predict the intended hand posture. Finally, the third stage processes the depth blob and the posture information to estimate the joints of the hand within the depth image.

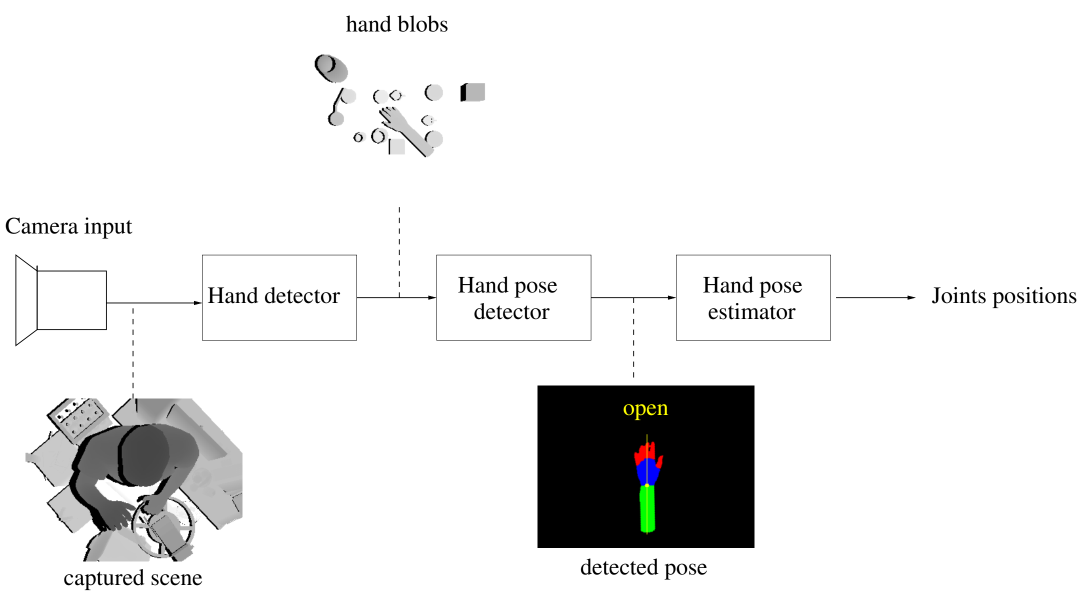


Figure 1.1: General diagram of global concept

This thesis focuses on the second stage of the discussed concept. The proposed solution receives as input a set of blobs from the original scene containing a hand. Such blob is segmented using a random decision forest (RDF) for pixel classification in order to find the hand region and divide it into *arm*, *palm* and *fingers* sections. Then seven visual features are tested in order to classify the hand pose using a second RDF, which classifies the blob using a pre-defined hand postures: *open*, *pointing*, *fist* and *pinch*. A support vector machine (SVM) is used for comparison for the hand pose classification stage.

1.1 Objective and Thesis Organization

This thesis has as primary goal the design of a system for hand pose detection using only depth images from a top view. In order to achieve such goal the system must be capable of segment a hand blob from a depth image and divide it into *arm*, *palm* and *fingers* sections. For this purpose a random decision forest is used. Furthermore, the system must be capable of estimating the wrist position, required for the next stage on the global three-staged concept. Additionally, the system must compute a set of visual features and train a second RDF for the hand pose prediction. It must be capable of distinguish between four basic hand postures: *open*, *pointing*, *fist* and *pinch*.

This document is organized as follows: Chapter 2 provides a brief discussion of the related work regarding hand pose estimation with depth images. Chapter 3 presents the basic theoretic foundations related with random decision forests, support vector machines and used visual features. Chapter 4 presents the proposed solution. The suggested implementation is then evaluated and discussed in Chapter 5. Finally, Chapter 6 summarizes the main conclusions and outlines future work.

Chapter 2

Related works

There is a plethora of research work aiming the analysis and recognition of hands using computer vision, due to the range of possible applications. With the development of depth sensors at low cost such as the Microsoft Kinect© or the Intel Creative Gesture Camera©, the field of research involving depth images for hand analysis has gained momentum.

Several approaches have been proposed for hand gesture and pose recognition using depth images. Keskin et al. [17] present a pose estimation algorithm for commonly used hand postures. They use a random decision forest (RDF) for per pixel classification and the mean shift algorithm is used to estimate the position of the joints. Their results are demostrated with a American Sign Language (ASL) digits classifier using the joints estimation.

Kuznetsova et al. [18] use a multi-layered random forest (MLRF) and ensemble of shape (ESF) to classify 24 hand postures from ASL depth images. They extract the point cloud from the 2D images and use the MLRF to cluster it in the first layer. The second layer in the MLRF is used to assign the corresponding hand pose to the data set. In [9] a different approach is taken to estimate the hand shape from depth data. In this case a sequence of widths is extracted from the hand contour and axis of elongation in order to be compared with a pre-defined synthetic data set using several similarity measurements. The hand segmentation is done under the premise that the hand is the closest object to the camera, so a simple threshold function can be applied.

Ren et al. [23] show a hand posture classification system using hand contour estimation. The depth image is segmented using color information and under two main assumptions: the hand must be the closest object to the camera and the person must wear a wristband. Once the hand is extracted from the original image, the finger-earth mover's distance (FEMD) is calculated from the time series curve of the hand contour. This feature is then used for template matching classification. A similar method is proposed by Dominio et al. in [10]. A hand depth image is segmented using depth and color information in order to extract the palm and finger regions. Then a distance sequence feature is calculated from the fingers to the palm alongside with a curvature feature of the fingers. A support vector machine (SVM) is then used to classify the features.

Pugeault and Bowden [22] use random decision trees to classify 24 hand postures of the ASL captured with a Microsoft Kinect. They use OpenNI[1] to localize the hand position and establish a region of interest around it on the depth and color images. Then a Garbor filter is used to get a feature vector from depth and intensity data that is going to be used to train an RDF for classification.

In general several constraints are necessary in order to perform a correct hand pose detection. Some of them rely in the proximity of the hand to the sensor and the point of view of the camera, restricting the hand position to be the closest object to the sensor from a front view. Other authors encourage, besides the depth information, the use of complementary color information for segmentation and feature extraction purposes.

Chapter 3

Foundations

This master's thesis focuses on the recognition of static hand poses from a top view using depth images. Some specific topics such as random decision forests, support vector machines among others are used in the implementation. This chapter aims to the description of those methods.

Section 3.1 provides a brief introduction to decision trees for classification, including the training and prediction processes. In section 3.2 the concept of decision tree is extended to random decision forest. Section 3.3 provides an insight in the theory of support vector machines. Finally section 3.4 explains the basics of geometric moments to present the Hu moments.

3.1 Decision Trees

As stated in [6], a decision tree can be defined as a special type of graph composed of nodes and edges as shown in figure 3.1 for a binary tree. A tree starts from an upper root node and splits up downwards into several internal nodes until it ends up at the terminal nodes commonly called leaf nodes in the Nth-level (where N is the tree's depth). The simplest arrangement of nodes for a decision tree is called binary decision tree, where each node splits up in at most two children: left and right.

Each internal node evaluates the incoming data based on a set of features and decision rules. The result of the decision rule in the current node will split the data into two groups: those samples that meet the decision criteria and those that do not.

In general a decision tree uses weak split functions to accomplish more complex or strong splitting processes in a similar way as boosting[12] does, leading to good generalization capabilities. This thesis focuses on decision trees for classification purposes. For a more detailed description of decision trees see Criminisi and Shotton [6].

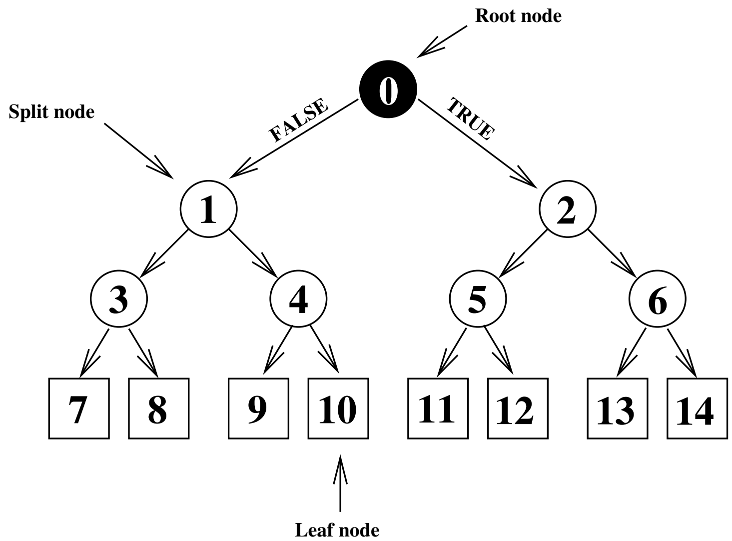


Figure 3.1: Graphical description of a binary decision tree (depth = 4)

3.1.1 Training Process

As described previously, a decision tree starts its training process from the root node splitting up the incoming data set into two groups: left and right. This splitting process continues on each node until it reaches a terminal node. This way each leaf node l_i allocates a distribution function $p(c|l_i(x))$ for each class label c and the input data x. Different estimators can be implemented in order to interpret such distribution function, being the most common method to select the class with the highest probability.

The training process can be summarized as an iterative process, executed for each node in the tree as follows[28, 6]:

1. Select a set of *data points*. Each data point corresponds to an specific sample within the entire data set.

2. Compute a set of features $\mathbf{v} = \{x_1, x_2, ..., x_d\}$ on each data point with $\mathbf{v} \in \mathbb{R}_d$. Ideally the dimension d for each feature space can be infinite, however in practice a subset of the whole features space is used instead.

3. Once the features are calculated for the data points, a test function $\tau(\theta)$ with $\theta \in \Theta$ is used to split the set in left and right samples. θ represents the parameter set of the test function belonging to the entire parameter space Θ.

4. The main goal of each node in the tree is to split the data in the *best* possible way taking the labeled samples as a reference, where *best* is defined in terms of an *objective function*. The goal of the objective function is to provide a quantitative measure of the splitting capability of the test function within the node and this way select the best set of parameters for the test function in the current node.

5. Once the best split function has been selected the resulting left and right data sets are passed to the next layer nodes and the process is applied recursively.

6. Each leaf node stores a probability distribution $p(c|\mathbf{v})$, based on the total of data points that reach the node and the corresponding label. This way each leaf provides a measure of the probability for a data point with a features set $\mathbf{v}$ to belong to an specific class c.

7. The training finishes until a stop criterion is met.

From the algorithm described above four key concepts have direct impact in the training process:

a) **Feature function**: the features used commonly for decision trees have the peculiarity of being weak compared with the complexity of the classification performed by the entire structure. They are completely dependent of the application and the data set used. For example Shotton et al. in [25] use the difference between two pixels in the neighborhood of the current point as the feature function for body parts detection in depth images. Waldvogel in [28] uses the difference of two regions in the neighborhood of the pixel as the feature function in order to create a per-pixel classifier using depth and color images.

b) **Split function**: The test function τ works on the computed features based on a set of split parameters θ and takes a decision about which branch the sample belongs to, such that

$$\tau(\mathbf{v}, \theta_t) \rightarrow \{true, false\} \qquad \forall \theta_t \in \Theta \tag{3.1}$$

Commonly the test function is a simple threshold value that can be set on advance or can be randomly selected.

c) **Objective function**: Also called score function, it evaluates each test function and helps to determinate the best split criteria for the current node. One of the most frequently used objective functions is the information gain. For a binary decision tree the information gain is defined as[6]

$$I = H(S) - \sum_{i \in \{L,R\}} \frac{S^i}{S} H(S^i) \tag{3.2}$$

where S is the data set and S_L, S_R are the splitted samples for each branch, H is the entropy in the corresponding set. The information gain is a variant of the Kullback-Leibler distance[14] providing a measurement between the data set entropy before the split and after. Lower entropy in the branched sets mean less uncertainty between the samples that conforms each set.

In short, the analysis of the objective function I can be seen as a maximization problem at each note jth for the data points in the node S_j and the specific split parameters set θ as follows

$$\theta_j = \underset{\theta}{\operatorname{argmax}}\, I(S_k, \theta) \tag{3.3}$$

d) **Stop criteria**: Growing full trees can be possible but impractical since it leads to over-fitted trees with low or zero generalization capability. Is for this reason that having an adequate stop criterion is a critical aspect of the training process. Diverse stopping criteria can be applied to a decision tree, being the most common the depth or number of levels of the tree. Alternatively other criteria can be used such as establishing a threshold for the objective function, determining the level of entropy in the leaf nodes. On the other hand the number of samples per node can be limited as a stop criterion avoiding the tree to overfit.

An interesting property of decision trees is that the chronological order of calculating node splits does not influence the decision tree structure[28], this gives rise to variants in the training process in regards with the order of nodes generation. Two main algorithms are widely used: depth-first training and breadth-first training. The former grows each child recursively until the stop criterion hold before it moves to the next adjacent node. In the case of breadth-first method all nodes on the same level are trained before moving to the next level. Figure 3.2 depicts both training methods, where numbers represent the order on which the algorithm processes the nodes and the dashed nodes represent future nodes to be processed.

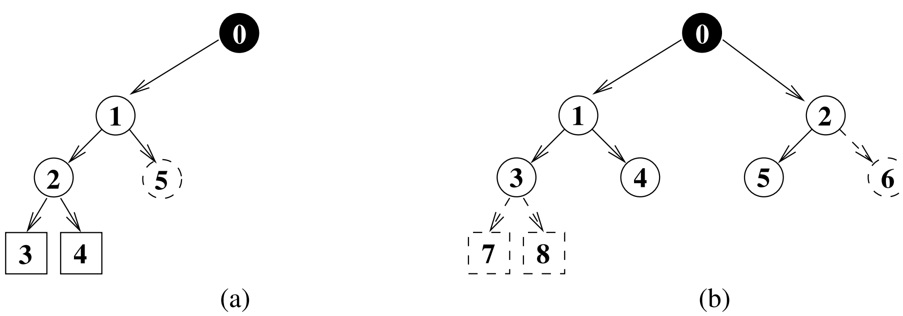


Figure 3.2: Decision tree training algorithms. (a) Depth-first method. (b) Breadth-first method.

When the training process has finished, three data sets are obtained that describe the tree entirely:

1. The best split functions for each node.
2. The tree structure.
3. The probability distribution for each leaf node.

The major drawback of decision trees is its training time, which can be considerable when dealing with big data sets. This disadvantage has encouraged the research of acceleration methods for the training process such as in [28].

3.1.2 Prediction Process

The prediction process starts in the same way as the training, splitting up the data set from the root node downwards to reach the leaf nodes.

From the training process each node was associated with an specific test function that minimize the entropy. When predicting new unseen data, the tree computes the corresponding features and splits it according to the training information. Since this process is only executed on the nodes involved on each split calculation the algorithm does not have to got through all the nodes within the tree. Figure 3.3 exemplifies the prediction path (in red) for a data point within a decision tree, it only requires the calculation of three nodes to reach a terminal node (represented with squares).

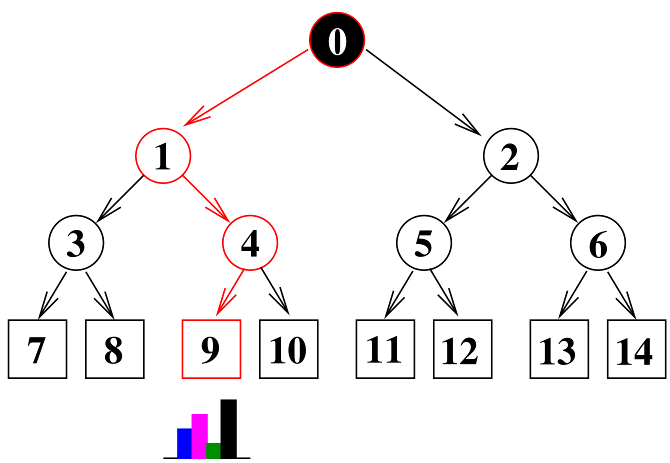


Figure 3.3: Prediction process for a decision tree. The processing path of a data point is marked in red.

Once a data point reaches the leaf node, the class probability distribution on that node allows to predict the most probable class for such data point. Different leaf predictors can be implemented. A common prediction model used is the Maximum A-Posteriori (MAP) estimate[6], described as

$$c^* = \underset{c}{\operatorname{argmax}}\, p(c, \mathbf{v}) \tag{3.4}$$

The structure of decision trees allows to reach the leaves in just N evaluations of test functions, which is suitable for real-time capable implementations.

3.2 Random Decision Forests

A Random Decision Forest (RDF) is an ensemble learning method constructed by a set of random decision trees as shown in figure 3.4 for an RDF comprised of 3 decision trees with 4 levels each one.

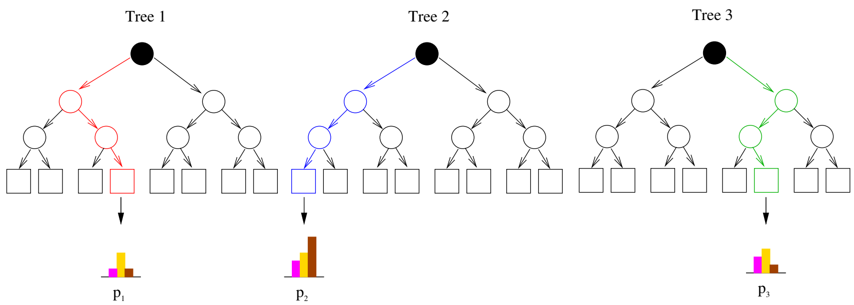


Figure 3.4: Graphical description of random decision forest ($T = 3$ and $depth = 4$). The 3 prediction paths are marked in red, blue and green producing 3 corresponding probability distributions.

In an RDF each tree is trained independently as described in section 3.1.1. The prediction is computed for each data point in all trees producing a set of T different probability distributions (where T is the forest size) as depicted in figure 3.4.

Several prediction models are used to determinate the final output of the forest, being the most common the calculation of the average probability distribution as follows[6]

$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v}) \tag{3.5}$$

The training and testing process in an RDF is achieved independently for each tree within the forest. This characteristic provides high parallelism leading to very efficient software implementations.

3.2.1 Randomness Model

Randomization is done in the training phase with the goal to improve the generalization capability of the classifier. There are two commonly used methods to inject randomness[6]:

1. Bagging
2. Randomized node optimization

Bagging is commonly used when the data set is small, thus an overfit of the tree is possible. The main idea in bagging is, given a finite data set S, a randomly selected subset S_t is chosen from the original data set for each tree t. Having a different random training set for each tree leads to the generation of different training parameter avoiding specialization.

Randomized node optimization (RNO) consists of training each node with a different random subset of split parameters Θ_t taken from the whole parameter space Θ, varying the set of possible tests performed on the features for each node. The more data points in Θ are contained in Θ_t (such that $\Theta_t \approx \Theta$), the more similar become the trees in the ensemble and the lower randomness is achieved, decreasing the generalization of the forest.

Randomness methods are not mutual exclusive which means that they can be implemented together in regards to the application requirements.

Summarizing, a set of six key parameters have a direct effect over the RDF training process as well as its prediction capabilities[6]:

1. The tree depth D
2. The size of the forest T
3. The objective function I for the training process
4. The feature function
5. The test function τ
6. The randomness methods injected

3.3 Support Vector Machines

Support Vector Machines (SVM), also known as support vector networks, are supervised learning methods for pattern recognition commonly used for binary classification applications[2, 16]. SVMs start from the basic concept of hyperplane separation. Consider a feature set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_i\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ (where d is the dimension of each data point); it may be possible to find an hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ that splits the data points in regards to their class labels.

Figure 3.5 shows an example of plane separation for two different hyperplanes h_1, h_2. In both cases each hyperplane separates the feature points as expected according with the corresponding labels and in fact, there are more possible options for hyperplanes that splits up the feature space correctly. In order to determinate the “best” split case a new concept is introduced: *margin*. The margin provides information about the minimal distance between a feature point $\mathbf{x}_i$ and the hyperplane's surface as shown in figure 3.5.

SVM considers the hyperplane selection as a maximization problem looking for the hyperplane that provides the largest margin between the samples of each class. This problem is reduced to maximize the distance of the two closest feature points to the hyperplane as shown in figure 3.5 for the feature points x_1 and x_2. Besides h_1 and h_2 splits up the

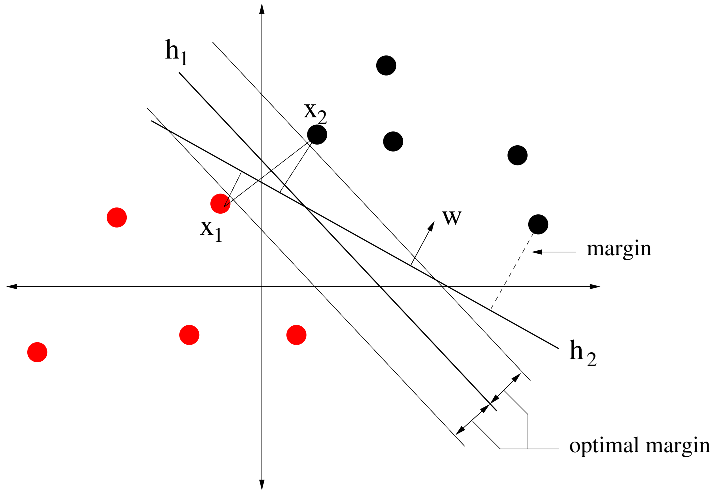


Figure 3.5: Features space separation using hyperplanes.

feature space correctly, h_1 provides the maximal margin between feature points x_1, x_2. Due to the roll of x_1 and x_2 in defining the optimal hyperplane for the feature separation they are called *support vectors*.

The split function is based on the hyperplane parameters and defined as

$$f(\mathbf{x}) = (\mathbf{x} \cdot \mathbf{w}) + b \tag{3.6}$$

such that $f(\mathbf{x}) \in \{-1, 1\}$, separating each class based on its sign.

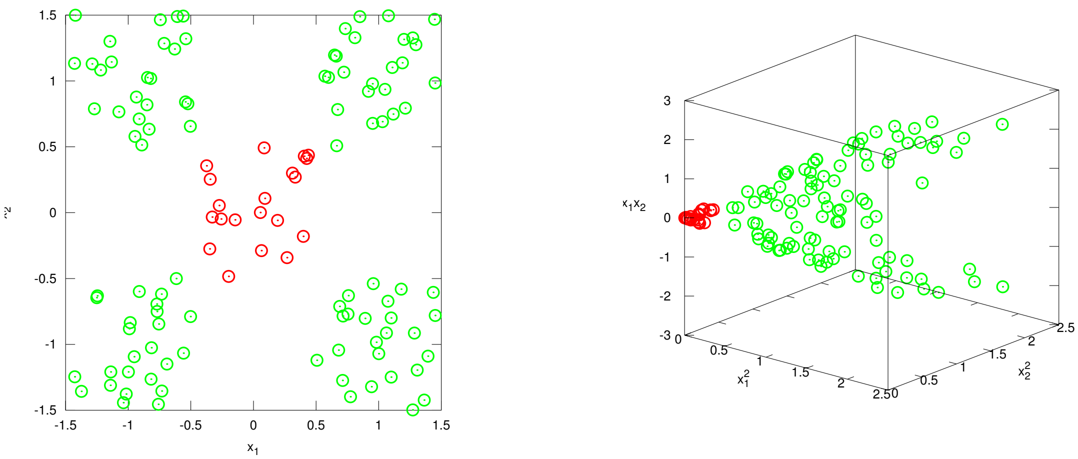
At this point it has been assumed that the feature set used is linearly separable; however this is not true in most of the cases. For example consider the features set in figure 3.6(a). There is not hyperplane capable to divide the feature set successfully. In those cases a nonlinear mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$ is applied, mapping the original features set S to a potentially much higher dimensional feature space $\mathcal{F}$. Increasing the dimensionality of the feature space will increase the probability of a linear separation of the feature set. As an example, let the mapping function ϕ be defined for the feature set in figure 3.6(a) as

$$\phi(\mathbf{x}) := (x_i^2, x_j^2, x_i x_j) \qquad \forall\ \mathbf{x} = (x_i, x_j) \tag{3.7}$$

Figure 3.6(b) shows the resulting feature space $\mathcal{F}$. The feature space dimensionality has increased to $d = 3$ but now the feature set is easily separable using an hyperplane.

Computing higher dimensional feature spaces provides a computational challenge due to the addition of complexity to the algorithm in real applications where the feature space $\mathcal{F}$ can be highly dimensional. In such cases explicitly computing scalar product operations between elements within the feature space becomes unfeasible. In order to overcome this limitation the use of *kernel functions* is widely extended.

A kernel function k is defined as



(a) (b)

Figure 3.6: Feature space mapping using kernels. a) Original 2D data set not directly separable through an hyperplane. b) Mapped data set using $\phi(\mathbf{x}) = \mathbf{z} = \{x_i^2, x_j^2, x_i x_j\}$ for $\mathbf{x} = \{x_i, x_j\}$.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{3.8}$$

so that there is not need to compute the mapping function $\phi(\mathbf{x})$ to calculate the scalar product between two feature points.

Different specific kernel functions have been defined such that (3.8) is met. Some of the most common kernels are shown in table 3.1[20]. The correct kernel function to use depends of the feature set and the final application.

Table 3.1: Common kernel functions used for SVM

Kernel	Definition
Gaussian RBF	$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) := \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|}{c}\right)$
Polynomial	$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) := ((\mathbf{x}_i \cdot \mathbf{x}_j) + \theta)^d$
Sigmoidal	$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) := \tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \theta)$
inv. multiquadric	$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) := \frac{1}{\sqrt{\|\mathbf{x}_i - \mathbf{x}_j\|^2 + c^2}}$

For a detailed description of SVM and their extensions to multiple class problems refer to [2, 20, 16].

3.4 Hu Moments in Image Processing

This section explains the basics of geometric moments in image processing in order to introduce the Hu moments.

3.4.1 Geometric Moments

The geometric moment M of order $(p+q)$ for a two-dimensional function $f(x, y)$ is defined as[19]

$$m_{pq} = \int_X \int_Y x^p y^q f(x, y) dx dy \tag{3.9}$$

where p, q are integers in the interval of $[0, \infty[$ and X, Y are the value ranges for (x, y) tuples. In the case of digital images (3.9) is replaced by[13]

$$m_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} x^p y^p f(x, y) \tag{3.10}$$

where $M \times N$ is the size of the image.

The entire moments sequence $\{m_{pq}\}$ uniquely describes an image $f(x, y)$ and in the same way, an image $f(x, y)$ is uniquely determined by an specific moments sequence. However this fact requires the computation of an infinite number of moments which is impossible in practice. It is possible to select an specific subset of moments that describes in an unambiguous manner an image for a specific application.

The lower order moments describe a set of fundamental geometric properties of the image function $f(x, y)$. Some of them are summarized in the table 3.2.

3.4.2 Moment Invariants

First introduced by Hu in [15], there are seven moment invariants or Hu moments useful to describe a two-dimensional image as follows:

$$\begin{aligned}
\phi_1 &= \mu_{20} + \mu_{02} && (3.11)\\
\phi_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 && (3.12)\\
\phi_3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 && (3.13)\\
\phi_4 &= (\mu_{30} - \mu_{12})^2 + (\mu_{21} - \mu_{03})^2 && (3.14)\\
\phi_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] && (3.15)\\
&\quad +(3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] &&\\
\phi_6 &= (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] && (3.16)\\
&\quad +4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{30}) &&\\
\phi_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] && (3.17)\\
&\quad -(\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] &&
\end{aligned}$$

Table 3.2: Geometric properties of a two-dimensional digital image based on its moments

Property	Definition	Description
Central Moments	$\mu_{pq} = \sum^M \sum^N (x-\bar{x})^p (y-\bar{y})^q f(x,y)$	With $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$ being the coordinates of the center of mass for a 2D image. The central moments of an image are invariant to translation.
Mass and area	$\mathrm{m}_{00} = \sum^M \sum^N f(x,y)$	Represents the total area of the image.
Center of Mass	$\begin{cases} m_{10} = \sum^M \sum^N x f(x,y) \\ m_{01} = \sum^M \sum^N y f(x,y) \end{cases}$	Represents the center of mass of an image with coordinates $(\bar{x}, \bar{y})$.
Orientations	$\{m_{02}, m_{11}, m_{20}\}$	Also known as moments of inertia, can be used to determinate the orientation of an image providing information about the angle of the main axes.

This set of moments are invariant to translation, scaling and rotation. Mirroring rotation is also present with a sign change in ϕ_7.

Chapter 4

Recognition of Static Hand Poses from a Top View

The main aim of this thesis is to identify a specific set of poses of a hand from a top view using depth images. Various visual features and classifiers are tested and its performance is compared.

This chapter presents the proposed system. Section 4.1 provides a general description of the algorithm implemented. Section 4.2 describes the data set used in the training process. A brief overview of the segmentation algorithm using RDF follows in Section 4.3. Section 4.4 introduces the visual features used for the pose classification stage and Section 4.5 provides a detailed description of the classifiers for pose recognition.

4.1 System Architecture

Figure 4.1 shows the general diagram of the system. The implementation is split in two processing phases:

1. **PHASE 1:** This phase is intended to the recognition of the hand and its main components (arm, palm and fingers) from a noisy environment such as a table with multiple objects. For this stage a modified version of Curfil [28] is used to train a random decision forest (RDF) based on depth images only. The resulting labeled blobs provide an estimation of the hand location and are feed to the next stage for the hand pose recognition.

2. **PHASE 2:** Once the hand is segmented the labeled blobs are used for feature extraction before sending them to the hand pose classifier. Various features are extracted from the labeled blobs and used to train a support vector machine (SVM) classifier and an RDF to compare their performance. The final result is the estimated hand pose.

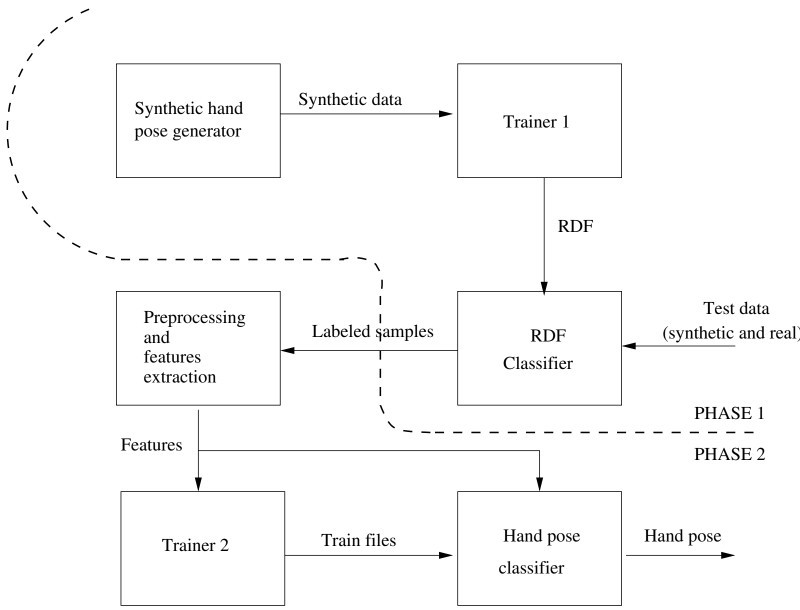


Figure 4.1: General diagram of the system developed

Additionally the hand wrist is located as part of the process as a key point for further stages of hand pose analysis.

4.2 The Data Set

The dataset is based on a series of images synthetically generated simulating the Kinect© depth sensor. For this thesis four hand poses were used as shown in table 4.1. Alongside the depth information, ground truth masks are generated that label the arm (green), palm (blue) and the fingers (red).

The camera is set from a top view. In all training sets, several background objects are added to the scene in order to let the RDF identify the hand correctly among noisy elements as shown in table 4.1. The objects are randomly located within the scene removing any location constrains, i.e., objects within the scene can be closer to the front plane than the hand and can be partially occluded behind it. The scene sets the background plane at 1882 mm and the hand Z position varies from 900 mm to 1500 mm rotating in its XYZ dimensions in predefined ranges ensuring the generated poses are physically possible, leading to a total of 13728 images per class.

The images are rendered based on a model of LibHand [27] but modified to include objects in the background as well as shadows within the image.

Each hand is rendered with and without simulated IR shadow to allow the RDF to ignore the shadows caused by the displacement between the source of the IR pattern and the IR camera. Figure 4.2 shows an example of the images generated.

Finally, in order to test the effect of the depth noise in the classification process two

Table 4.1: Hand classes, depth image and ground truth masks

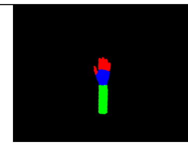
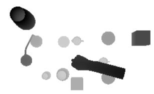
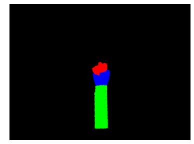
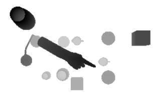
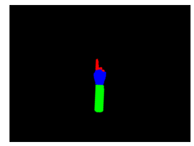
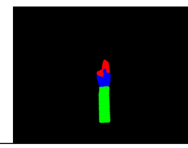
Class Name	Depth image	4-Labels Mask
Open		
Fist		
Pointing		
Pinch		

Figure 4.2: Generated depth image without shadow (left) and with shadows (right)

variants of the training set are generated: a first data set contains the synthetic images described above and the second has added noise in the depth images as encountered in the real depth sensors. The noise is added using the noise model given in [4] for Z dimension only as follows:

$$\sigma(i,j,z) = \beta_1 j^2 + \beta_2 i^2 + \beta_3 z^2 + \beta_4 ji + \beta_5 iz + \beta_6 zj + \beta_7 j + \beta_8 i + \beta_9 z + \beta_{10} \qquad (4.1)$$

where β_x are constant coefficients for Z direction, i and j are the region indices as calculated in [4] and $\sigma(i,j,k)$ is the standard deviation of the gaussian distribution modeling the noise.

Table 4.2 summarizes the different data sets generated.

Due to memory limitations the images in the data set generated have a size of 160×120 pixels.

Table 4.2: Data sets (50% images with shadow)

Data Set #	Noise (% images within the set)	# Images per Class
1	100%	13728
2	0%	13728
3	50%	27456

4.3 Phase 1: Pixel Segmentation using RDF

In order to detect the hand in the data set a segmentation process is required. For this reason an RDF is used to learn the hand components in the training data and perform a per pixel classification process.

The training process is achieved using a modified version of Curfil[28], removing any dependencies to RGB information and using exclusively the depth images. Each tree is independently trained using breadth-first order. The feature implemented is based on the difference of the average value between two regions on the image in the neighborhood of the pixel that is being treated. Figure 4.3 shows the feature calculation for a pixel q and offsets o_1 and o_2. Each region is defined by its w_i and h_i values corresponding to the width and height dimensions respectively.

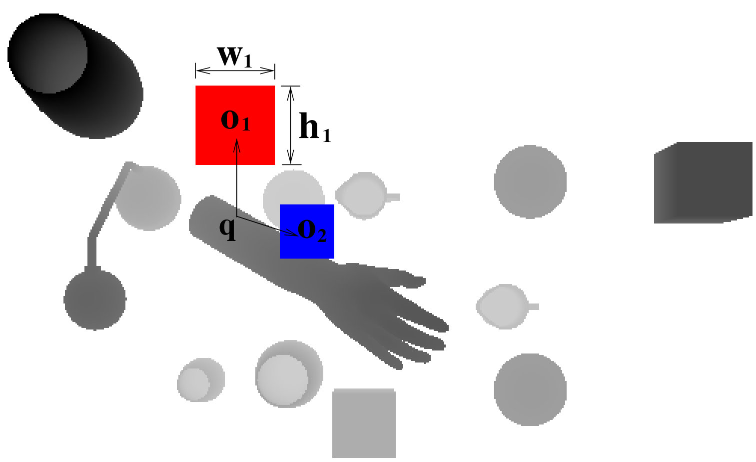


Figure 4.3: Example of Curfil's feature calculation in a depth image

The offset o_i and dimension values w_i and h_i, are randomly selected in the training process and normalized with respect the depth value of the query pixel q. The final image feature function is defined as[28]

$$f_\theta := \frac{1}{|R_1(q)|}\sum_{p\in R_1} d(p) - \frac{1}{|R_2(q)|}\sum_{p\in R_2} d(p) \tag{4.2}$$

where $R_i, i \in \{1,2\}$ is the region defined by the relative offset o_i and the dimension parameters w_i, h_i normalized by the depth of the query pixel $d(q)$ as follows:

$$R_i(q) := R\left(q + \frac{o_i}{d(q)}, \frac{w_i}{d(q)}, \frac{h_i}{d(q)}\right) \forall i \in \{1,2\} \tag{4.3}$$

Curfil implements the information gain as its objective function for impurity estimation. The information gain is defined as the difference of Shannon entropy of the class distribution D in the parent node and the weighted sum of Shannon entropies in class distribution in each child over the set of classes C[28] as follows

$$I_C(D, D_{LEFT}, D_{RIGHT}) := H_C(D) - \frac{|D_{LEFT}|H_C(D_{LEFT}) + |D_{RIGHT}|H_C(D_{RIGHT})}{|D|} \tag{4.4}$$

where the entropy over the classes H_C for the current distribution D is defined as

$$H_C(D) := -\sum_{c\in C} p(c|D)\log_2(p(c|D)) \tag{4.5}$$

Each tree is trained using randomly selected images from the data set, preserving the same class occurrence rate to avoid training bias.

The resulting RDFs generated from the training process are used to classify any incoming images and localize the hand and its components, such labeled images are feed to the next stage for the hand pose estimation.

4.4 Visual Features for Hand Pose Classification

In order to classify the labeled blob a set of visual features are extracted. Figure 4.4 shows the general preprocessing flow.

First, the labeled blobs are processed using morphological operations in order to minimize any missclassification in the pixels and a binary mask is extracted identifying the hand region. Second, the eigenvectors and eigenvalues of the mask are calculated to get the principal axes of the hand, defined with twice the standard deviation of the corresponding axis as follows:

$$\text{major axis length} = 2\sigma_{major} \tag{4.6}$$

$$\text{minor axis length} = 2\sigma_{minor} \tag{4.7}$$

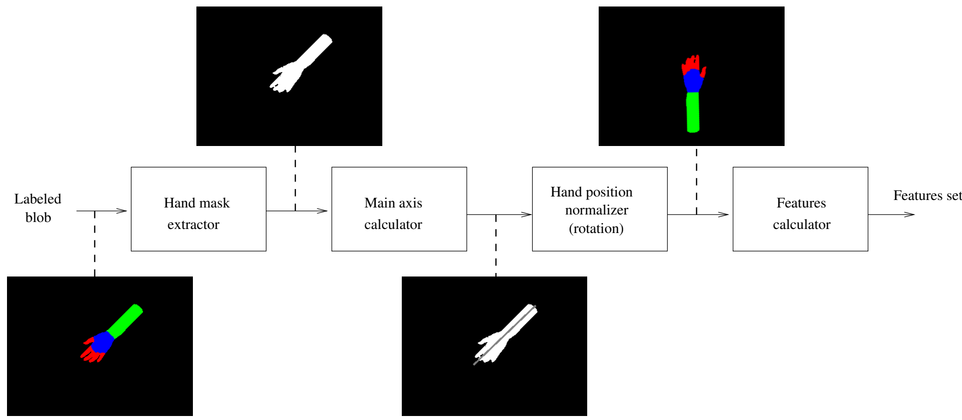


Figure 4.4: Features extraction process for hand pose classification

The hand position is then normalized by rotating its main axis at 90°. The hand is always rotated with the fingers at the top of the image based on the colors histogram at the ends of the main axis.

Once the hand position is normalized, the next step computes the features for the classification. Four features are used:

1. **Entire mask labels histogram:** The first feature corresponds to the labels histogram of the entire blob based on the mask region, ignoring any information about background pixels:

$$h(c) = \frac{1}{N} \sum_{i \in S} p_c(i) \qquad , c \in C := \{palm, arm, fingers\} \tag{4.8}$$

   with

$$p_c(i) = \begin{cases} 1 & L(i) = c \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

   Each bin c in the histogram h is calculated as the sum of all the pixels assigned to such class c in the points of the blob S. Two different values for the parameter N are used for testing: $N \in \{1, \sum_{c \in C} h(c)\}$.

2. **Hand ROI histogram:** For this feature additional information is extracted from the labeled images: the location of the hand's wrist, found by the analysis of the main axis histogram and its boundary between the palm and the arm as shown in the figure 4.5.

   Once the wrist is found, a rectangular region is calculated with one of its sides defined by the wrist point and the upper end of the hand's main axis; the other side corresponds to the mask's minor axis. The resulting region is depicted in figure 4.6.

   With the rectangular ROI defined, the calculation of the feature corresponds to the histogram of labels considering the background labels and ignoring the arm labels.

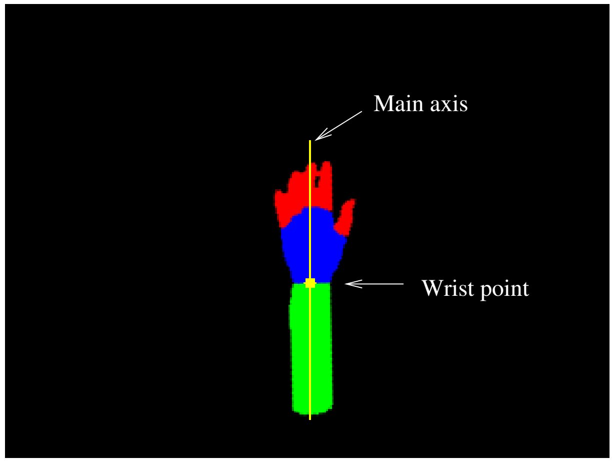


Figure 4.5: Location of the hand's wrist

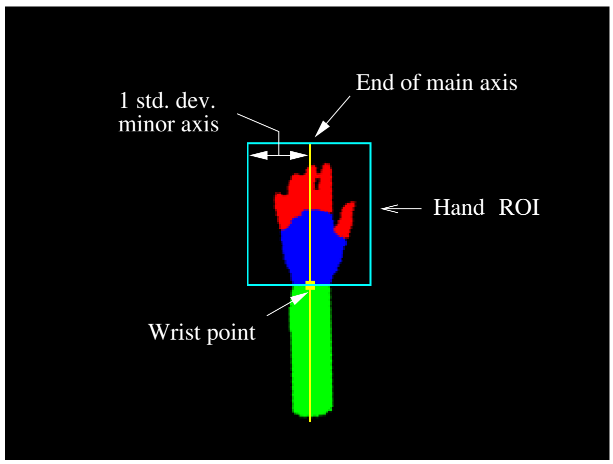


Figure 4.6: Hand ROI for histogram calculation

Arm labels doesn't provide any additional information about the shape of the hand since they belong to the arm itself which is independent of any hand's pose. The background pixels on the other side provide information about the hand pose in a similar way to a convex hull calculation. This way the histogram is calculated as follows

$$h(c) = \frac{1}{N} \sum_{i \in S_R} p_c(i) \qquad , c \in C := \{fingers, palm, background\} \tag{4.10}$$

where S_R corresponds to the points of all the pixels that belongs to the rectangular area and $N = \sum_{c \in C} h(c)$.

3. **Hand ROI quadrants histogram:** This feature splits the main rectangular ROI of the hand, calculated in the previous point, into four quadrants and calculates the histogram for the palm, fingers and background labels among the four quadrants. Figure 4.7 shown the four quadrants calculated in the image.

   Splitting the main rectangle into four regions provides information about the changes

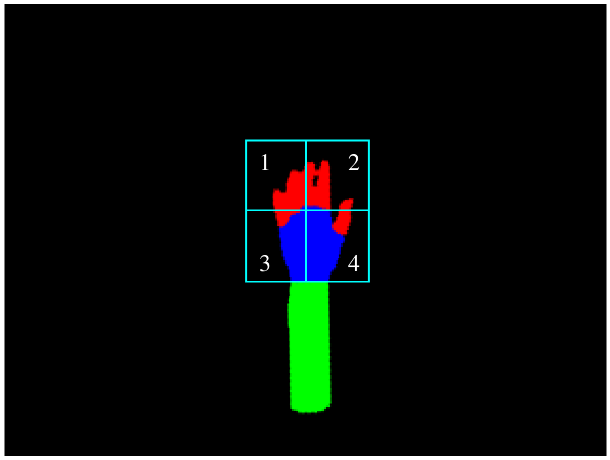


Figure 4.7: Hand ROI quadrants for histogram calculation

of the hand pose on each quadrant independently, being able to capture discriminative patterns for each hand region.

Two variants of the feature are calculated for testing, the first one calculates a three element histogram for each quadrant $q_j, j \in \{1, 2, 3, 4\}$ as follows:

$$h_{q_j}(c) = \frac{1}{N} \sum_{i \in S_Q} p_c(i) \qquad , c \in C := \{fingers, palm, background\} \tag{4.11}$$

where S_Q corresponds to the points of all the pixels that belongs to the quadrant area and $N = \sum_{c \in C} h_{q_j}(c)$.

The second variant of the feature unifies the background labels count into a single entry for the entire histogram, reducing the feature's dimension from 12 to 9. This way the histogram calculation is expressed as

$$\begin{cases} h_{q_j}(c) = \frac{1}{\sum_{c \in C} h_{q_j}(c)} \sum_{i \in S_Q} p_c(i) & \text{for } c \in \{fingers, palm\} \\ h(c) = \frac{1}{\sum_{c \in C} h(c)} \sum_{i \in S_R} p_c(i) & \text{for } c = background \end{cases} \tag{4.12}$$

4. **Hu Moments:** Hu moments are used as features in a similar way than in [5] but considering each hand region independently. The first step is to separate the hand palm and fingers regions from the rotated image as shown in figure 4.8. The hand regions are extracted using a region growing algorithm and then passed to the Hu calculator.

Table 4.3 summarizes the features described and shows their corresponding dimensions.

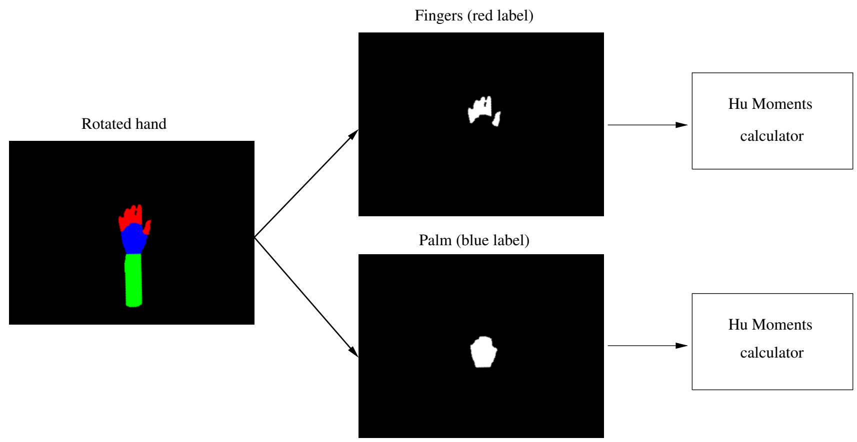


Figure 4.8: Hand regions extraction and Hu moments calculation from rotated image

Table 4.3: Features for hand pose classification

#	Feature	# dimensions
1	Full image histogram	3
2	ROI histogram	3
3	ROI histogram (denormalized)	3
4	ROI quadrants histogram	12
5	ROI quadrants histogram (reduced)	9
6	Hu moments	14

4.5 Phase 2: Gesture Classification

The gesture classification stage is intended to provide an estimation of the hand pose based on the features described in the previous section. Two classifiers are tested in order to compare their accuracy in the classification process; a random decision forest (RDF) and a support vector machine (SVM) are selected for this phase.

Each classifier is trained with the features extracted from the previous section and used to estimate the hand pose.

The training is done with randomly selected blobs to avoid training bias.

Chapter 5

Experimental results

In this chapter a set of experimental results on the proposed system are shown. Section 5.1 describes the different data sets used in the experiments. Section 5.2 provides a summary of the segmentation results produced by the RDF classifier in the first stage of the system. A brief analysis of the wrist location results is provided in section 5.3 as a key point for further stages of the system. Finally, section 5.4 provides a detailed analysis of the results from the second stage of the system, aimed to the classification of the hand pose. An RDF classifier is tested and its prediction capability is compared against a classic SVM classifier. At last, the RDF classifier for pose estimation is exposed to ideally labeled hand blobs in order to determinate the classification performance of the second stage assuming an ideal segmentation.

5.1 Data set

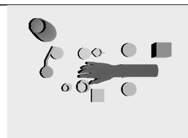
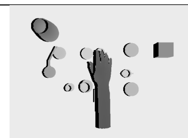
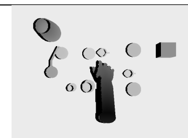
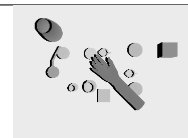
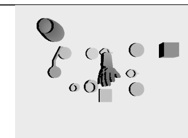
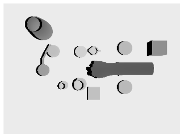
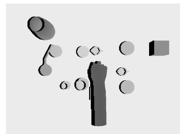
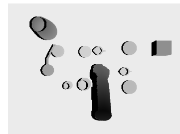
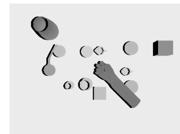
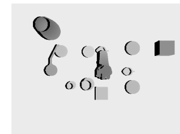
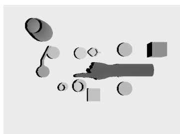
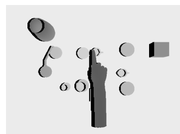
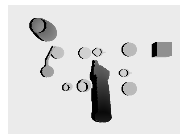
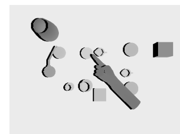
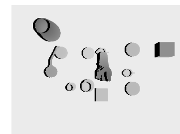
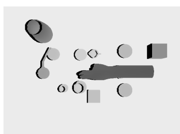
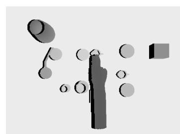
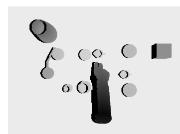
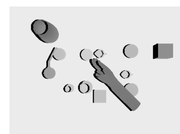
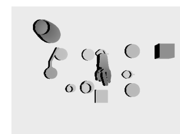
The generation process of the data set in use was provided in chapter 4.2. The experiments along this chapter are based on a set of three different data sets presented in section 4.2 and shown in table 5.1 for convenience. Each data set differs on the amount of images with noise within the entire set. All data sets contain images rendered with shadows and without them, to improve shadows immunity. A total of 4 hand poses are used: *open*, *pointing*, *fist* and *pinch*, as depicted in table 5.2 for several rendering location/rotation variations.

Table 5.1: Data sets used for experimental results (50% images with shadow)

Data Set #	Noise (% images within the set)	# Images per Class
1	100%	13728
2	0%	13728
3	50%	27456

The identifiers 1, 2 and 3 are used along all the chapter to identify the corresponding data set used on each experiment.

Table 5.2: Hand pose samples at different location/rotation values

Class Name	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Open					
Fist					
Pointing					
Pinch					

A total of 10000 randomly selected images from each data set are used to train the first stage of the system aimed to the hand segmentation. An additional batch of 10000 are then used for testing that stage and feed to the hand pose classification stage to train the RDF and SVM classifiers. Finally a 5000 images set is used to validate the classifiers of such stage.

5.2 Segmentation of Hand Regions

This section presents a set of experimental results aimed to the analysis of the hand segmentation stage of the system presented in section 4.3. Three RDF classifiers are trained with the data sets introduced in section 5.1 in order to validate the effect of the noise within the training images. These results correspond to the classification output for each individual pixel, where no information of neighbor classifications is taken into account. Furthermore, detailed analysis on the prediction capabilities of the RDF is performed.

Each RDF is trained with the set of parameters listed in table 5.3. No parameters optimization has been performed since such topic is out of the temporal scope of this thesis.

The classifiers are identified as follows:

- **RDF A**: RDF trained with data set 1, specially focused on the classification of blobs without noise.

Table 5.3: RDF configuration for hand segmentation

Parameter	Value
Max depth	16
Size (trees)	5
Samples per image	600
Number of thresholds	1000
Features per node	1500
Feature max box edge size (region size)	10
Feature max box offset (box radius)	30
Train set size	10000
Test set size	10000

- **RDF B**: RDF trained with data set 2, specially focused on the effect of noise in the prediction capability.
- **RDF C**: RDF trained with data set 3. This test is aimed to test the effect of training the classifier with samples with noise and without noise.

The accuracy of each classifier has been measured quantitatively and summarized in tables 5.4, 5.5 and 5.6, which correspond to the confusion matrices for the classifiers A, B and C respectively. C_0, C_1, C_2 and C_3 are the *background*, *fingers*, *arm* and *palm* classes, respectively. The prediction results between the three random forests is very similar, with only a maximum variation of 7% around the mean. Since the background pixels comprise the majority of the region on each blob its accuracy in each matrix is near the 100%; however the classification of the pixels belonging to the hand parts presents more confusion, being the *fingers* the class with the larger confusion, classifying around 46% of the samples as *background* and *palm*.

Table 5.4: Confusion matrix for RDF A trained with depth images without noise. (Data set size = 10000 with 0% images without noise)

Confussion Matrix		**Predictions**			
		C0	**C1**	**C2**	**C3**
Classes	**C0**	0.997	0.001	0.002	0.000
	C1	0.208	0.538	0.020	0.233
	C2	0.081	0.002	0.904	0.013
	C3	0.075	0.032	0.033	0.860

Since the *background* class may bias the measurement of accuracy for each classifier, as an additional quantitative measurement here it is proposed to take the *background* class out of the statistics. Defining the average per-class accuracy as the average of the diagonal

Table 5.5: Confusion matrix for RDF B trained with noisy depth images. (Data set size = 10000 with 100% images with noise)

Confussion Matrix		**Predictions**			
		C0	**C1**	**C2**	**C3**
Classes	**C0**	0.997	0.001	0.002	0.000
	C1	0.191	0.473	0.037	0.300
	C2	0.076	0.002	0.904	0.018
	C3	0.073	0.028	0.054	0.845

Table 5.6: Confusion matrix for RDF C trained with mixed depth images. (Data set size = 10000 with 100% images with noise)

Confussion Matrix		**Predictions**			
		C0	**C1**	**C2**	**C3**
Classes	**C0**	0.997	0.001	0.002	0.000
	C1	0.197	0.513	0.030	0.261
	C2	0.080	0.004	0.897	0.019
	C3	0.077	0.041	0.043	0.839

of the confusion matrix[24] and omitting the *background* predictions leads to the results in table 5.7 for the accuracy rates for the classes C_1, C_2 and C_3 and the total per-class accuracy. It is possible to notice an slight reduction in the prediction accuracy when noise is injected, decreasing in up to 4.6% the average prediction for the classifier B and 2.8% for the classifier C.

Table 5.7: Class accuracy omitting background class prediction statistics

Classifier	C_1 accuracy	C_2 accuracy	C_3 accuracy	Average accuracy
A	0.680	0.984	0.930	0.865
B	0.584	0.978	0.912	0.825
C	0.638	0.975	0.910	0.841

The average classification accuracy for *arm* is 97.9% and 91.7% for the *palm*, being the *fingers* the class with the lower average prediction with 63.4%, which coincides with the results obtained from the confusion matrices. The *fingers* class presents its lower prediction accuracy for the RDF B with 58.4%.

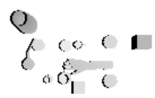
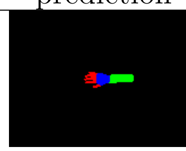
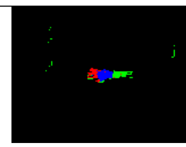
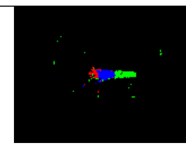
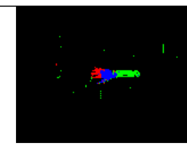
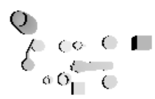
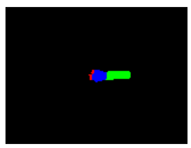
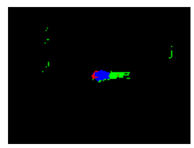
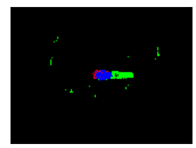
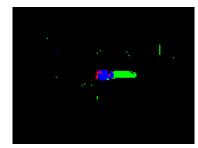
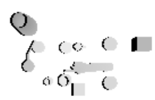
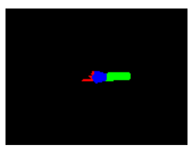
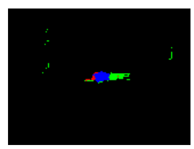
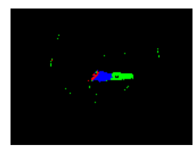
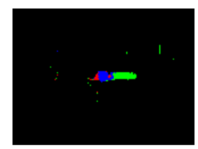
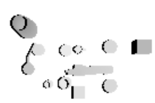
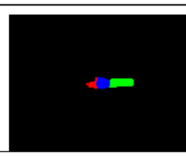
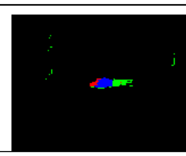
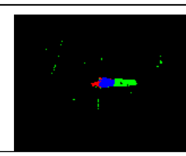
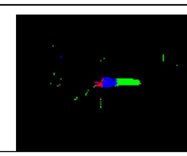
Two interesting results are obtained from the confusion matrices in tables 5.4, 5.5 and 5.6 and the accuracy statistics in table 5.7. First, the RDF capability to ignore or minimize the effect of the noise, presenting a close average accuracy in the three tests performed.

Second, the better accuracy of the RDF C compared with the RDF B. Training the RDF with 50% of noisy images helped the forest to increase its accuracy in around 1.8%, improving the RDF immunity to the noise.

Since the RDF C presents the best average accuracy with noisy images, improving considerably the pixel prediction for the *fingers*, it is selected as the reference classifier along the remaining sections of this thesis.

Table 5.8 shows an example of the prediction results for each classifier.

Table 5.8: Experimental hand segmentation results for RDF A, B and C (Using training parameters from table 5.3)

Class Name	Depth image	Ideal prediction	RDF A	RDF B	RDF C
Open					
Fist					
Pointing					
Pinch					

5.3 Wrist Location

One of the key processing stages within the system is the estimation of the wrist point in the labeled blobs provided by the segmentation phase.

In order to get a quantitative measurement of the wrist estimation, the relative wrist point distance is calculated, i.e., the Euclidean distance between the ideal wrist point and the estimated wrist point normalized by the standard deviation of the minor axis of the arm, as follows

$$\text{Relative Distance} = \frac{\sqrt{(x_{ideal} - x_{exp})^2 + (y_{ideal} - y_{exp})^2}}{\sigma_{arm_{minor}}} \tag{5.1}$$

With the above definition a relative distance lower or equal than 1 would mean that the difference between the ideal and estimated wrist points lies within a circle with center (x_{ideal}, y_{ideal}) and with radius $\sigma_{arm_{minor}}$.

Table 5.9 shows the relative wrist distance measurements for each class and for the entire data set, providing the mean and standard deviation. Additionally, the percentage of samples with a relative wrist distance lower than 1 is provided. The average mean relative distance is 0.509 with standard deviation values around 0.337, this means that most of the wrist points calculated provide a good estimation of the real wrist point location. This result is confirmed by looking at the percentage of samples within the radius $\sigma_{arm_{minor}}$ which in average corresponds to 93.63%.

Table 5.9: Relative wrist point distance measurements for RDF predicted images (Data set size = 10000)

Measument	Open	Pointing	Fist	Pinch	Average
Mean	0.517	0.508	0.501	0.509	0.509
Std. Dev.	0.366	0.315	0.336	0.330	0.337
Samples within radius (%)	92.84	93.56	94.48	93.64	93.63

In order to determinate whether the wrist estimation is out of the hand region, the relative distance has been calculated for the x and y coordinates separately as listed in table 5.10. The x coordinates have an average relative distance of 0.284 with 99.95% of the samples within the radius. In the other hand the y coordinates present an average relative distance of 0.426 with 94.32% of the samples within the radius.

Table 5.10: Relative wrist point distance for x and y coordinates (Data set size = 10000)

Measument	Open	Pointing	Fist	Pinch	Average
Mean X	0.272	0.289	0.284	0.292	0.284
Std. Dev. X	0.198	0.216	0.199	0.202	0.204
% Samples Rel$\Delta X < 1$	100	99.86	100	99.95	99.95
Mean Y	0.449	0.425	0.421	0.411	0.426
Std. Dev. Y	0.389	0.328	0.357	0.357	0.359
% Samples Rel$\Delta Y < 1$	93.35	94.67	94.77	94.49	94.32

Since the wrist point is estimated with the blobs with normalized position, the results in table 5.10 imply that the 99.95% of the points fall into the arm region with only major location differences in the y coordinate. This major variation in the y coordinate can be attributed to the misclassification of the *palm* class with the *arm* in the segmentation stage.

5.4 Hand Pose Classification

This section presents the analysis of the hand pose classification stage of the system. The RDF and SVM classifiers are tested and their performance is discussed alongside with the experimental results.

The features used for the hand pose estimation are listed in table 5.11 (for more details about their definition see section 4.4). Feature 7 has been included for analysis purposes and it is composed as the union of all the previous features, the main goal of this feature is to evaluate the discriminatory capabilities of each classifier among all the range of possible features to use.

Table 5.11: Experimental visual features used for hand pose classification

#	Feature	# dimensions
1	Full image histogram	3
2	ROI histogram	3
3	ROI histogram (denormalized)	3
4	ROI quadrants histogram	12
5	ROI quadrants histogram (reduced)	9
6	Hu moments	14
7	Complete feature set	45

5.4.1 Classification Using RDF

The training parameters used for the RDF are shown in table 5.12.

Table 5.12: RDF configuration used for hand pose classification

Parameter	Value
Max depth	[10,25]
Size (trees)	[10,150]
Objective function	entropy
Train data set size	10000
Test data set size	5000

The RDF depth is varied from 10 to 25 levels in 5 units increments and the forest size is tested within a range of 10 to 150 trees with 10 units increments for each feature. Those configurations are used in order to evaluate the best match for the data set in use. Figure 5.1 presents the classification error for each RDF (trained using the corresponding feature) for different depth/size configurations. The classification error is mostly constant for each feature, being the RDF trained with the complete set of features the one presenting the lower classification error, close to 10%. The ROI quadrants histogram and its option with dimensions reduced present a classification error close to 20%, being it the isolated feature that provide best classification results. On the other hand the Hu moments provide the worst classification error for all the RDF configurations.

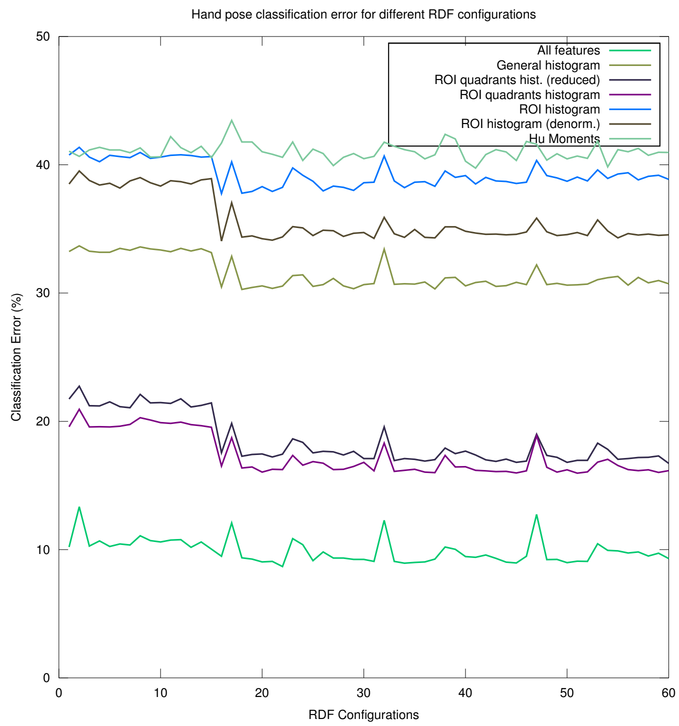


Figure 5.1: Classification error vs RDF configuration for different features

Table 5.13 shows the minimum classification error (i.e. the maximum average per-class accuracy) for each feature and the corresponding depth and size values. The best result is emphasized with a gray shadow, provided by the entire set of features using an RDF of depth 15 and size 150, with an average per-class accuracy of 91%.

Table 5.13: Best classification error and average per-class accuracy using all features

Feature	RDF Depth	RDF Size (trees)	Min. Classification Error (%)	Average Per-class accuracy
All Features	15	150	8.680	0.91
General histogram	15	110	30.280	0.70
ROI Quadrants hist. (red.)	25	90	16.720	0.83
ROI Quadrants hist.	25	140	15.960	0.84
ROI hist. (denormalized)	15	100	34.060	0.66
ROI histogram	15	100	37.760	0.62
Hu Moments	20	50	39.740	0.60

Figure 5.2 shows the classification error curves for the RDF trained with the entire features set at a constant depth value and varying the RDF size. Depth values above 15 levels behave in a similar way with a minimum classification error around 8.87%. This fact is presented in more detail in table 5.14 which presents the best classification error and average per-class accuracy values for each RDF depth. For depth values above 15 levels the best average per-class accuracy is 91% and it only varies on the number of trees used.

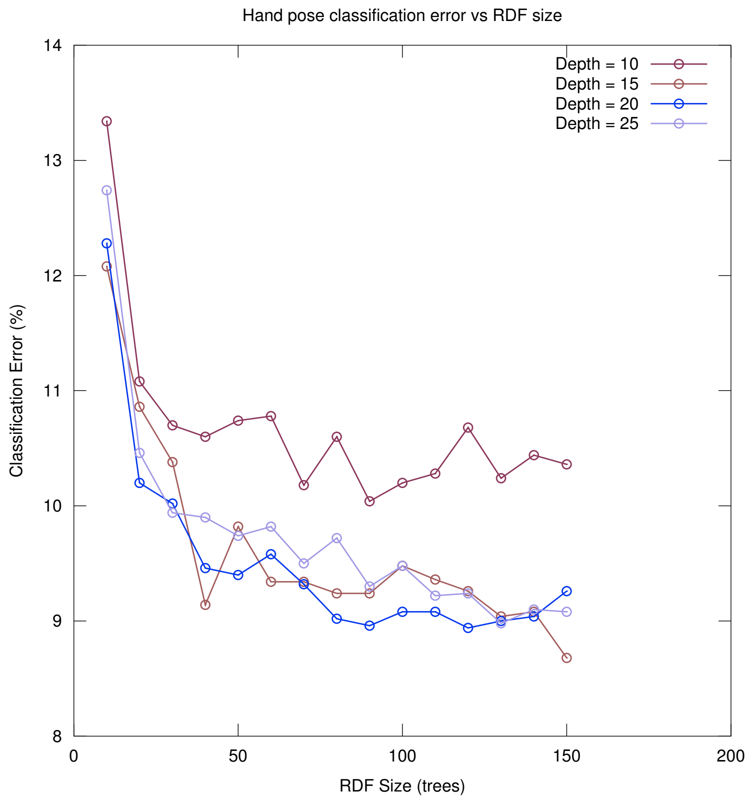


Figure 5.2: Classification error vs RDF size for different depth values using the entire features set

Table 5.14: Best classification error and average per-class accuracy using all features

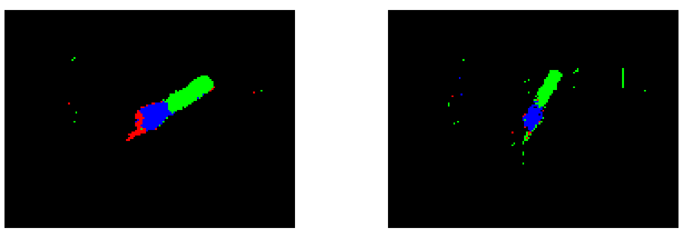
RDF Depth	RDF Size (trees)	Min. Classification Error (%)	Average Per-class accuracy
10	90	10.040	0.90
15	150	8.680	0.91
20	120	8.940	0.91
25	130	8.980	0.91

Table 5.15 shows the confusion matrix for the RDF trained with a depth of 25 levels and a size of 130 trees, which presented the best prediction accuracy. In this setup the *open* hand label presents the higher prediction accuracy (with a 96%). This result can be associated to the fact that this class presents the most distinguishable visual features

among the others. Additionally, *fist* and *pinch* classes present an accuracy rate higher than 91% leaving the *pointing* class at last with 84%. As it is expected the *pinch* and *pointing* classes present a mutual confusion of around 7% due to their physical similarity. An interesting result is the confusion presented between the *pointing* and *fist* labels, probably caused by the misclassification of the index finger pixels, leading to a segmented blob similar to a *fist*, as shown in the experimental results in figure 5.3. Despite the segmentation of the blob in figure 5.3 a) provides a correct pixel assignment for the index finger, in the figure 5.3 b) the index pixels were classified to the *arm* class which leads to a poor set of visual descriptors for the pose estimator.

Table 5.15: Confusion matrix for hand classification using the complete set of features (depth = 25, RDF size = 130)

Confussion Matrix		**Predictions**			
		Open	**Pointing**	**Fist**	**Pinch**
Classes	**Open**	0.968	0.013	0.00	0.019
	Pointing	0.008	0.839	0.081	0.072
	Fist	0.002	0.074	0.918	0.007
	Pinch	0013	0.070	0.002	0.916



(a) (b)

Figure 5.3: Misclassification of *pointing* class pixels. a) Index finger correctly segmented. b) Index finger misclassified pixels

Similar to boosting, random decision forests provide useful information about the importance of each feature component within the training process. This is a interesting feature commonly used to select and reduce the size of a feature set. Table 5.16 provides a summary of the features importance rates obtained from the RDF analysed. Features based on the ROI quadrants analysis presents the higher importance percentage within the range of 23-33%. Features such as the Hu moments with 14 components provides only a 9.501% of importance, with 10 components with a 0%; being the first moment for palm and the first and second moments for the fingers the ones with higher importance. Additionally, in all the histogram based features the *finger* component provides the higher percentage of information for the pose classification, which can be related to the fact that this visual feature is the most dynamic between different hand postures.

The results listed in table 5.16 provide an insight of the main components within the features set that are meaningful for the classification process. This information can be used to reduce the size of the feature set, removing all the components that provide 0% of information or close and just computing the main components.

Table 5.16: Features importance percentage for an RDF trained with all the features sets (depth = 15, size = 120)

Feature	Component	Component Importance (%)	Feature Importance (%)
General Histogram	Finger	6.61	9.482
	Arm	1.08	
	Palm	1.76	
	NA	0.03	
ROI Quadrants Histogram (Reduced)	Finger 1	2.99	23.938
	Palm 1	1.61	
	Finger 2	7.41	
	Palm 2	3.00	
	Finger 3	1.33	
	Palm 3	0.91	
	Finger 4	3.47	
	Palm 4	1.20	
	Background	2.02	
ROI Quadrants Histogram	Finger 1	2.84	32.618
	Palm 1	1.83	
	Background 1	3.42	
	Finger 2	7.04	
	Palm 2	3.82	
	Background 2	4.19	
	Finger 3	1.82	
	Palm 3	0.90	
	Background 3	0.96	
	Finger 4	3.24	
	Palm 4	1.37	
	Background 4	1.19	
ROI Histogram	Finger	7.91	12.501
	Palm	2.49	
	Background	2.10	
ROI Histogram (Denormalized)	Finger	7.79	11.960
	Palm	1.39	
	Background	2.78	
Hu Moments	Palm: 1	1.48	9.501
	Palm: 2	0.04	
	Palm: 3	0.00	
	Palm: 4	0.00	
	Palm: 5	0.00	
	Palm: 6	0.00	
	Palm: 7	0.00	
	Fingers: 1	5.76	
	Fingers: 2	2.22	
	Fingers: 3	0.00	
	Fingers: 4	0.00	
	Fingers: 5	0.00	
	Fingers: 6	0.00	
	Fingers: 7	0.00	

In the next section SVM classification is analysed in order provide a comparison point of the result presented with a classic ensemble.

5.4.2 Classification Using SVM

In this section a set of support vector machines are trained to perform the hand pose classification from the labeled blobs provided by the segmentation stage using the same set of features listed in table 5.11. The goal of this test is to compare the RDF performance with a classic and widely used classification algorithm.

The SVM is used with a Gaussian RBF kernel with a Gaussian spread $\gamma = 0.0001$ and a cost parameter $C = 5.275$. Such parameters were selected by using a grid analysis and finding the best combination within the ranges shown in table 5.17.

Table 5.17: Parameters range used for SVM grid analysis

Parameter	Range
γ	[0.1, 0.0002]
C	[1, 2000000]

Table 5.18 summarizes the classification error and average per-class accuracy for each feature. As expected from the result obtained from the RDF analysis, the SVM presented the best classification accuracy when the entire set of features is used as input to the classifier. However, classification accuracy is much lower for the SVM than the obtained using RDF with a minimum classification error of 24.94%. It is interesting to see how the Hu moments present the worst classification accuracy among the features set, coinciding with the results obtained for the RDF classifier.

Table 5.18: Classification error and average per-class accuracy for SVM classifier ($\gamma = 0.0001, C = 5.275$)

Feature	Classification Error (%)	Average Per-class accuracy
All Features	24.94	0.75
General histogram	35.74	0.64
ROI Quadrants hist. (red.)	50.14	0.50
ROI Quadrants hist.	50.08	0.50
ROI hist. (denormalized)	39.26	0.61
ROI histogram	56.96	0.43
Hu Moments	72.28	0.28

5.4.3 Classification of Ideal Predicted Blobs using RDF

It has been demonstrated the superiority of the random forests for the hand pose classification compared with support vector machines classifiers, for the features set proposed. In this section it is analysed the classification capability of the proposed features using random forests and an ideal set of predicted blob labels. Separating the hand blob segmentation error from the classification process allows to measure the prediction capabilities of the system in a controlled environment.

The RDF is trained with the same parameters shown in table 5.12 for each of the features proposed. Figure 5.4 depicts the classification error based on different depth and size settings for the RDF on each feature. In contrast with the results obtained in figure 5.1, all the features but the Hu moments got a classification error under 5%. Table 5.19 provides a summary of the best candidates for classification for each feature. All random forests presented an average per-class accuracy above 90% with the Hu moments based RDF being the less accurate with 91%.

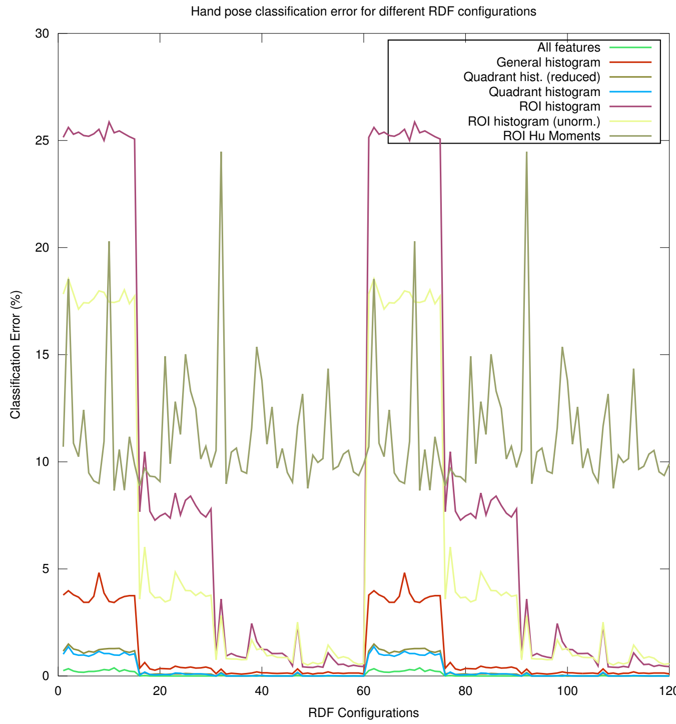


Figure 5.4: Classification error vs RDF configuration for different features with ideal segmentation of hand blobs

Table 5.19: Best classification error and average per-class accuracy using all features

Feature	RDF Depth	RDF Size (trees)	Min. Classification Error (%)	Average Per-class accuracy
All Features	15	100	0.00	1.00
General histogram	20	110	0.090	1.00
ROI Quadrants hist. (red.)	20	100	0.000	1.00
ROI Quadrants hist.	20	100	0.000	1.00
ROI hist. (denormalized)	25	120	0.510	0.99
ROI histogram	25	130	0.400	1.00
Hu Moments	10	50	8.660	0.91

Table 5.20 summarizes the features importances. Again the ROI quadrants features provide the most discriminative feature with about 58% of importance rate. Despite the fact that Hu moments just reach a 26% of importance in the classification process, it mainly belongs to the first moment of the palm and the first and second moments for the fingers, coinciding with the results obtained in section 5.4.1.

Table 5.20: Features importance percentage for an RDF trained with all the features sets with ideal segmentation (depth = 15, size = 100)

Feature	Component	Component Importance (%)	Feature Importance (%)
General Histogram	Finger	3.72	4.234
	Arm	0.12	
	Palm	0.39	
	NA	0.00	
ROI Quadrants Histogram (Reduced)	Finger 1	4.32	24.013
	Palm 1	2.06	
	Finger 2	3.92	
	Palm 2	6.70	
	Finger 3	1.72	
	Palm 3	1.25	
	Finger 4	2.57	
	Palm 4	0.78	
	Background	0.69	
ROI Quadrants Histogram	Fingers 1	5.19	34.650
	Palm 1	3.55	
	Background 1	1.83	
	Fingers 2	4.53	
	Palm 2	5.94	
	Background 2	1.98	
	Fingers 3	2.92	
	Palm 3	2.10	
	Background 3	0.56	
	Fingers 4	4.18	
	Palm 4	1.20	
	Background 4	0.66	
ROI Histogram	Finger	4.00	7.542
	Palm	2.92	
	Background	0.63	
ROI Histogram (Denormalized)	RED	2.20	3.548
	Palm	0.27	
	Background	1.08	
Hu Moments	Palm: 1	7.28	26.012
	Palm: 2	0.24	
	Palm: 3	0.00	
	Palm: 4	0.00	
	Palm: 5	0.00	
	Palm: 6	0.00	
	Palm: 7	0.00	
	Fingers: 1	9.44	
	Fingers: 2	9.06	
	Fingers: 3	0.00	
	Fingers: 4	0.00	
	Fingers: 5	0.00	
	Fingers: 6	0.00	
	Fingers: 7	0.00	

The results presented demonstrate a high dependency of the features proposed with the accuracy of the segmentation of the hand blob, being highly discriminative when the blob information tends to an ideal prediction.

Chapter 6

Conclusions

This thesis presents a two staged architecture for the hand pose recognition of four hand postures (*open*, *pointing*, *fist* and *pinch*), using depth images in a top view perspective without any color information. The proposed design removes some of the classic constrains in hand recognition with depth sensors such as the hand relative position to other objects and verticality with respect to the camera. Additionally the proposed solution addresses some common error sources in IR sensors such as Gaussian noise and shadows.

Along the entire project a synthetic data set was used to perform the training and verification test. Such data set was generated based on LibHand hand model[27] and allowed the analysis of various characteristics of depth images independently, such as the effect of the depth noise in the training process.

The first processing stage was implemented using a modified version of Curfil[28] to train an RDF for per-pixel classification, identifying the arm, palm and fingers in the depth images. It was found that the addition of Gaussian noise in the depth images (based on Kinect's noise model[4] for Z direction) causes a decrement of just 4.6% in the classification accuracy, exposing a high level of immunity to Gaussian noise. Also, using a combination of depth images with 50% of the samples with noise for training increments the pixel classification accuracy of the RDF in about 1.8%, increasing the RDF immunity to noise.

It was possible to get a classification accuracy for the segmentation stage up to 84.1%, being the fingers the most difficult region to estimate with a 26% of confusion with the palm.

As part of the intermediate process, the wrist point of the hand was estimated for each blob, with a total average relative distance to the theoretical point of 0.509 ± 0.337; fitting the 93.63% of the calculated wrist points within the $\sigma_{armminor}$ radius around the theoretical point.

In the second stage of the system the RDF classification capability was compared with a classic SVM implementation for the prediction of the hand posture, based in a set of seven different visual features (see table 5.11) extracted from the segmented blobs. It was proved the higher capability of the random forests to learn from a set of features and provide an acceptable prediction rate compared with SVM, providing up to a 91% of average per-class accuracy against 75% of the SVM. Additionally it was demonstrated that pixel based features calculated from the ROI quadrants around the hand provide the higher discriminative information in the classification process. The Hu moments of the palm and fingers provide the lowers discriminative information presenting meaningful metrics only on their first and second moments.

In general it has been proved the capability of random decision forests for hand pose classification from depth images where no hand location constrains or color information is used. Additionally it has been possible to identify a set of simple visual features that provide high discriminatory information for hand pose estimation for hand blobs.

6.1 Future Work

It is left for future work to test the proposed solution under real conditions using real depth images for the prediction process. On the other hand, the major disadvantage of RDF is the required training times, which considerably limit the amount of points in the parameter space that can be explored. Tuning of such parameters is a task still to be performed to optimize the results obtained. Additionally, other RDF ensembles variants can be used in order to improve the classification accuracy and overall performance, such as the usage of multi-layered random forests (MLRF)[18] that improves memory usage and speed.

A new set of features can be explored for the hand pose classification stage, either providing more discriminative information about the hand ROI pixels arrangement or extracting information directly from the depth information. Contour features, such as fingers curvature or contour distance respect the wrist point, can easily be integrated based on the visual features already calculated within the system.

Bibliography

[1] OpenNI. URL http://www.openni.org.

[2] Dustin Boswell. Introduction to Support Vector Machines. pages 1–15, 2002. URL http://videolectures.net/epsrcws08_campbell_isvm/.

[3] CR Cameron. Hand tracking and visualization in a virtual reality simulation. *(SIEDS), 2011 IEEE*, 22904:127–132, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5876867.

[4] Benjamin Choo, Michael Landau, Michael DeVore, and Peter Beling. Statistical Analysis-Based Error Models for the Microsoft KinectTM Depth Sensor. *Sensors*, 14(9):17430–17450, 2014. URL http://www.mdpi.com/1424-8220/14/9/17430/.

[5] S Conseil, S Bourennane, and L Martin. Comparison of Fourier Descriptors and Hu Moments for Hand Posture Recognition . *EURASIP*, page 20, 2007.

[6] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. 2013.

[7] Arnaud Dapogny, Raoul De Charette, and Sotiris Manitsaris. Towards a Hand Skeletal Model for Depth Images Applied to Capture Music-like Finger Gestures. *10th Int. Symposium on Computer Music Multidisciplinary Research (CMMR'2013)*, 2013. URL http://hal-ensmp.archives-ouvertes.fr/hal-00875721$\delimiter"026E30F$nhttp://hal.archives-ouvertes.fr/hal-00875721/.

[8] Martin de La Gorce, David J Fleet, and Nikos Paragios. Model-Based 3D Hand Pose Estimation from Monocular Video. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1793–1805, February 2011. URL http://www.ncbi.nlm.nih.gov/pubmed/21339527.

[9] Paul Doliotis, Vassilis Athitsos, and Dimitrios Kosmopoulos. Hand Shape and 3D Pose Estimation using Depth Data from a Single Cluttered Frame. pages 1–11, 2012.

[10] Fabio Dominio, Mauro Donadeo, Giulio Marin, Pietro Zanuttigh, and Guido Maria Cortelazzo. Hand gesture recognition with depth data. In *Proceedings of the 4th ACM/IEEE International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Stream*, ARTEMIS '13, pages 9–16, New York, NY, USA, 2013. ACM. URL http://doi.acm.org/10.1145/2510650.2510651.

[11] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, October 2007. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314206002281http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.6351&rep=rep1&type=pdf.

[12] Yoav Freund, Robert E Schapire, Park Avenue, and Florham Park. A Short Introduction to Boosting. 14(5):771–780, 1999.

[13] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[15] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[16] Christian Igel. Machine Learning: Kernel-based Methods. pages 21–23, 2014. URL http://image.diku.dk/igel/teaching/KernelBasedMachineLearning.pdf.

[17] Cem Keskin, Furkan Kırac, Yunus Emre Kara, and Lale Akarun. Real Time Hand Pose Estimation using Depth Sensors. *IEEE International Conference on Computer Vision Workshops*, pages 1228–1234, 2011.

[18] Alina Kuznetsova, Laura Leal-Taixe, and Bodo Rosenhahn. Real-Time Sign Language Recognition Using a Consumer Depth Camera. *2013 IEEE International Conference on Computer Vision Workshops*, pages 83–90, December 2013. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6755883.

[19] Simon Xinmeng Liao. Image Analysis by Moments. Technical report, University of Manitoba, 1993.

[20] Klaus Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. *IEEE Transactions on Neural Networks*.

[21] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. *Procedings of the British Machine Vision Conference 2011*, pages 101.1–101.11, 2011. URL http://www.bmva.org/bmvc/2011/proceedings/paper101/index.htmlhttp://www.researchgate.net/publication/233950932_Efficient_model-based_3D_tracking_of_hand_articulations_using_Kinect/file/9fcfd50d72d96c01ca.pdf.

[22] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time ASL fingerspelling recognition. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1114–1119, 2011.

[23] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, page 1093, 2011. URL http://dl.acm.org/citation.cfm?doid=2072298.2071946.

[24] By Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-Time Human Pose Recognition in Parts from Single Depth Images. 2011.

[25] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*. IEEE, June 2011. URL http://research.microsoft.com/apps/pubs/default.aspx?id=145347.

[26] Danhang Tang, Tae-kyun Kim, and Tsz-Ho Yu. Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests. 2013.

[27] Marin Šarić. Libhand: A library for hand articulation, 2011. URL http://www.libhand.org/. Version 0.9.

[28] Benedikt Waldvogel. *Accelerating Random Forest on CPUs and GPUs for Object-Class Image Segmentation*. PhD thesis, 2013.

[29] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28:1, 2009.

[30] Yoshihiro Watanabe, Atsushi Matsutani, Takehiro Niikura, Takashi Komuro, and Masatoshi Ishikawa. High-speed estimation of multi-finger position and pose for input interface of the mobile devices. In *The 1st IEEE Global Conference on Consumer Electronics 2012*, pages 228–232. IEEE, October 2012. URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6379588.